Article ID: 1005-2429(2004)02-0107-09

Real-Time Fuzzy Obstacle Avoidance Using Directional Visual Perception

Huang Guoquan（黄国权） Rad A. B. Wong Y. K.

Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China

Abstract

This paper presents a novel vision-based obstacle avoidance approach for the Autonomous Mobile Robot (AMR) with a Pan-Tilt-Zoom (PTZ) camera as its only sensing modality. The approach combines the morphological closing operation based on Sobel Edge Detection Operation and the ($\mu - k\sigma$) thresholding technique to detect obstacles to soften the various lighting and ground floor effects. Both the morphology method and thresholding technique are computationally simple. The processing speed of the algorithm is fast enough to avoid some active obstacles. In addition, this approach takes into account the history obstacle effects on the current state. Fuzzy logic is used to control the behaviors of AMR as it navigates in the environment. All behaviors run concurrently and generate motor response solely based on vision perception. A priority based on subsumption coordinator selects the most appropriate response to direct the AMR away from obstacles. Validation of the proposed approach is done on a Pioneer 1 mobile robot.

Key words Fuzzy system; Obstacle avoidance; Edge detection; Autonomous mobile robot

Introduction

It is a fundamental problem for Autonomous Mobile Robots (AMRs) to avoid obstacles when navigating. When robots are used to replace human beings to do some boring and repeating work, such as to transport objects[1], there may be some obstacles scatted on its way, so that the robot must have the ability to avoid them to effectively fulfill the task. In the past, a variety of approaches for vision-based obstacle avoidance of AMR has been developed. Ohya et al.[2] used a model-based map for AMR navigation on a planned path. However, there are many different obstacles in the real-world, which if modeled might slow the processing of safe passage. This problem also limits the portability of the system into specific environment. We have the strong interest in building an algorithm that can be used in different indoor environments. Lorigo et al.[3] extended the obstacle avoidance by fusing three different independent vision modules. The method used the RGB, HSV, and brightness gradient of the image and performed well in most indoor and outdoor environments, but it did not consider the lighting variations, which maybe has an bad effect on the performance of AMR. Potts[4] mainly focused on the computation ability and simplified the obstacle detection and path planning. Miin et al.[5], however, combined the vision sensor and ultrasonic sensor like Ohya[2] to do the work. Maja et al.[6] used segmentation technique to segregate the ground from other fixtures; the algorithm was proved remarkably robust in various lighting conditions, but they assumed the ground shares the pixels with the objects. Other methods also exist; for example, Tanaka[7] did this work by observing walking people, and Ku et al.[8] adopted vehicle location estimation and quadratic classifier to accomplish the obstacle avoidance

Received 2004—02—25.

Foundation item The National Natural Sience Foundation of China (No. 623852).

Biographies Huang Guoquan (1979—), research assistant, g.q.huang@polyu.edu.hk; Rad A.B. (1955—), professor, doctor, eeabrad@polyu.deu, hk; Wong Y.K. (1955—), assistant professor, doctor, eeykwong@polyu.edu.hk.

in person following for vision-based autonomous land vehicle guidance.

In this paper, we present an approach that combines morphological closing operation and $(\mu - k\sigma)$ thresholding technique like that in Ref. [6], and thus lead to a robust vision-based navigation system softening the effects of lighting and ground limitation. Especially this approach also takes into account the effect of previous obstacles which may be not in the current view zone, just like that human can remember the obstacles they saw before.

It has widely been accepted to use the subsumption architecture to build the complex artificial intelligent system that provides a structured approach to combining simple behaviors to exemplify an artificial intelligence, which is normally called behavior-based artificial intelligence[9]. Robot behaviors can be implemented as a set of fuzzy rules which mimic expert knowledge in specific tasks to model the expert knowledge. Recently, many researchers proposed to use the fuzzy logic to implement auto-navigation for AMR[10-12]. Usually human do not need precise, numerical input information to make a decision, but they are able to perform highly adaptive and robust control. Fuzzy logic is known to be an organized method for dealing with imprecise data. A fuzzy system attempts to apply a human-like way of thinking, which is usually designed by interviewing an expert and formulating his implicit knowledge of the underlying process into a set of linguistic variables and fuzzy rules. For design of a behavior-based AMR, expert knowledge is also needed to plan and implement the desired behaviors. The behaviors usually emerge from implicit knowledge of the underlying process that can be converted into a set of linguistic variables and fuzzy rules. In this work, fuzzy logic is applied to the behavior-based AMR with a vision sensor only. The vision data becomes the input of the fuzzy logic system and the steering angle and translation velocity become the outputs. Thus, fuzzy obstacle avoidance is successfully implemented by formulating a set of fuzzy rules.

1 Design Specification

Note that the AMR's only goal is to navigate safely in an unstructured environment solely by a Pan-Tilt-Zoom (PTZ) camera, without target destination. Therefore, it is a local navigation problem[11].

1.1 Platform

As shown in Fig. 1, the robot used in this work is a Pioneer 1 mobile robot mounted with a PTZ camera, which is a Sony EVI-D30 with 12X optical zoom, high speed auto-focus lens and a wide angle len, a pan range of $\pm 95^\circ$ (at a maximum speed of 80 $(^\circ)/s$), and a tilt range of $\pm 20^\circ$ (at a maximum speed of 50 $(^\circ)/s$). Resolution of frame grabbed from camera video is 160×120 pixels with true color, and the maximum rate can reach 100 frames per second, which is fast enough to meet the real-time requirement.

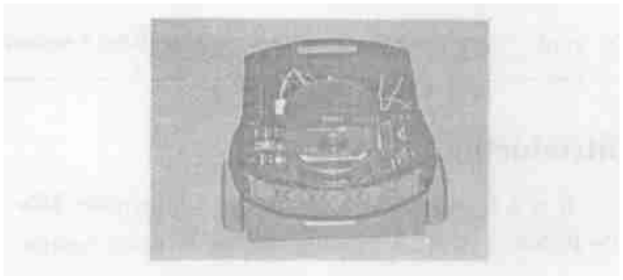

Fig. 1 Pioneer 1 mobile robot

1.2 Overall control architecture

Fig. 2 shows the control architecture of the robot system. The vision system is used to detect the obstacles, process the dynamic images real time, and provide accurate information about the obstacles. With the

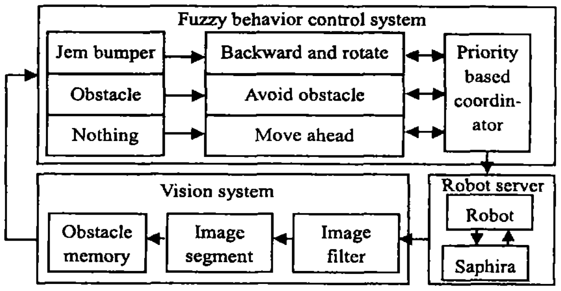

Fig. 2 Overall control architecture

information, AMR can accomplish obstacle avoidance through fuzzy behavior control system when navigating in its environment.

1.3 Assumption

With the assumption that all obstacles lie on the ground and that the robot is working on a flat surface, the ground-plane constraint[3] means that the objects close to the robot is represented at the lower level of the frame, while the objects far away are found at the top of the frame. The object found nearer, which cover the upper portion of the frame, is considered as a near object due to its attachment to the ground. The constraint transformation pair used by Lorigo[3], Miin[5] and Maja[6] is also used in this work.

2 Vision System

2.1 Image filter

The image filter takes in visual frames from the camera video, and extracts the original data from the images contaminated with noises. Due to the lighting variation and video wireless transmission and other unreliable problems, there must be much noises in the raw frame. So we need to filter these noises as much as possible before the vision data are used to direct the AMR. Hence, we adopt median filter to remove the impulse noise and neighborhood operation like Box operation and Gaussian operation to remove the random noise. After smoothing, we use Laplacian operation to sharpen the edge of the objects in the image, which is helpful for segmentation because our destination is to find the obstacles. See Fig. 3.

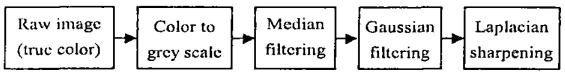

Fig. 3 Image filtering series

2.2 Image segmentation

Grey scale segmentation or thresholding is to convert a grey scale image $I(x, y)$ into a bi-value image $B(x, y)$[13]. The common way of this conversion is to select a threshold T:

$$B(x, y)=\begin{cases}0, & \text{if } I(x, y)<T,\\ 1, & \text{if } I(x, y)\geqslant T.\end{cases} \tag{1}$$

Therefore, the main problem of segmentation is how to determine the threshold value T. Though various thresholding techniques exist[6, 13], the robustness of these techniques to segment the ground in different environments should also be investigated. Many of these approaches assume that the distribution is bimodal, and this is acceptable in most indoor conditions. Ref. [6], however, assumes that the ground and the objects share pixels, which means that the distribution is rather even or symmetrical, but this will not work well in those bimodal or multimodal distribution environments. Here in this paper, the thresholding technique in Ref. [6] and the morphological closing based on Sobel edge detection are combined to segment the image, and hence can be suitable for more kinds of environments. Which method should be used is determined by analyzing the diagram of the filtered image at current state. An illustration of the image segmentation is shown in Fig. 4.

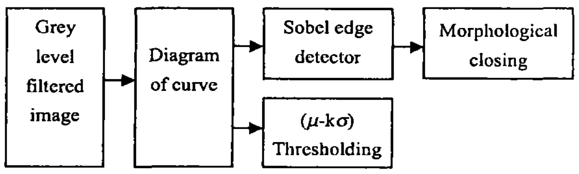

Fig. 4 Image segmentation

In the bimodal distribution conditions, we use Sobel edge detection operation, which is more simple in computation than Canny edge detection operation[14], and morphological closing operation to segment the gray scale image obtained from the image filter. The Sobel operator performs a 2-D spatial gradient measurement on an image. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. The Sobel edge detector uses a pair of 3×3 convolution, one estimating the gradient in the x-direction (G_x) and the other estimating the gradient in the y-direction (G_y). Then the magnitude of the gradient is calculated by

$$|G|=\sqrt{G_x^2+G_y^2}\ . \tag{2}$$

After detecting the edges, morphological closing is used to convert the edge image $E(x, y)$ into bi-value image $B(x, y)$. Morphological closing is to recover the initial shape of the image structures that have been dilated by eroding the dilated image[15]. The closing by a structuring element B is denoted by Φ_B and is defined as the dilation with a structuring element B followed by the erosion with the transposed structuring element $\hat{B}$:

$$\Phi_B=\varepsilon_{\hat{B}}\delta_B. \tag{3}$$

However, in the conditions where the ground shares pixels with the objects, an approach similar to that in Ref. [5] is adopted. Due to the pixel-sharing characteristic, we use the pixel value of ($\mu-k\sigma$) as the threshold value, which is computed by

$$\left.\begin{aligned} \mu &= \frac{\sum_{i=1}^{N} X_i}{N} \\ \sigma &= \sqrt{\frac{\sum_{i=1}^{N}(X_i-\mu)^2}{N-1}} \\ T &= \mu-K\sigma \end{aligned}\right\}, \tag{4}$$

where μ is mean value; σ is standard deviation; N is the number of pixels; X_i is intensity value in the image; and $K=1$ in this work. Although the output does not produce a perfect true positive rate, it is sufficient to generate safe passages. Fig. 5 and Fig. 6 show the sampled multimodal and single-modal images in vision system, reapectively.

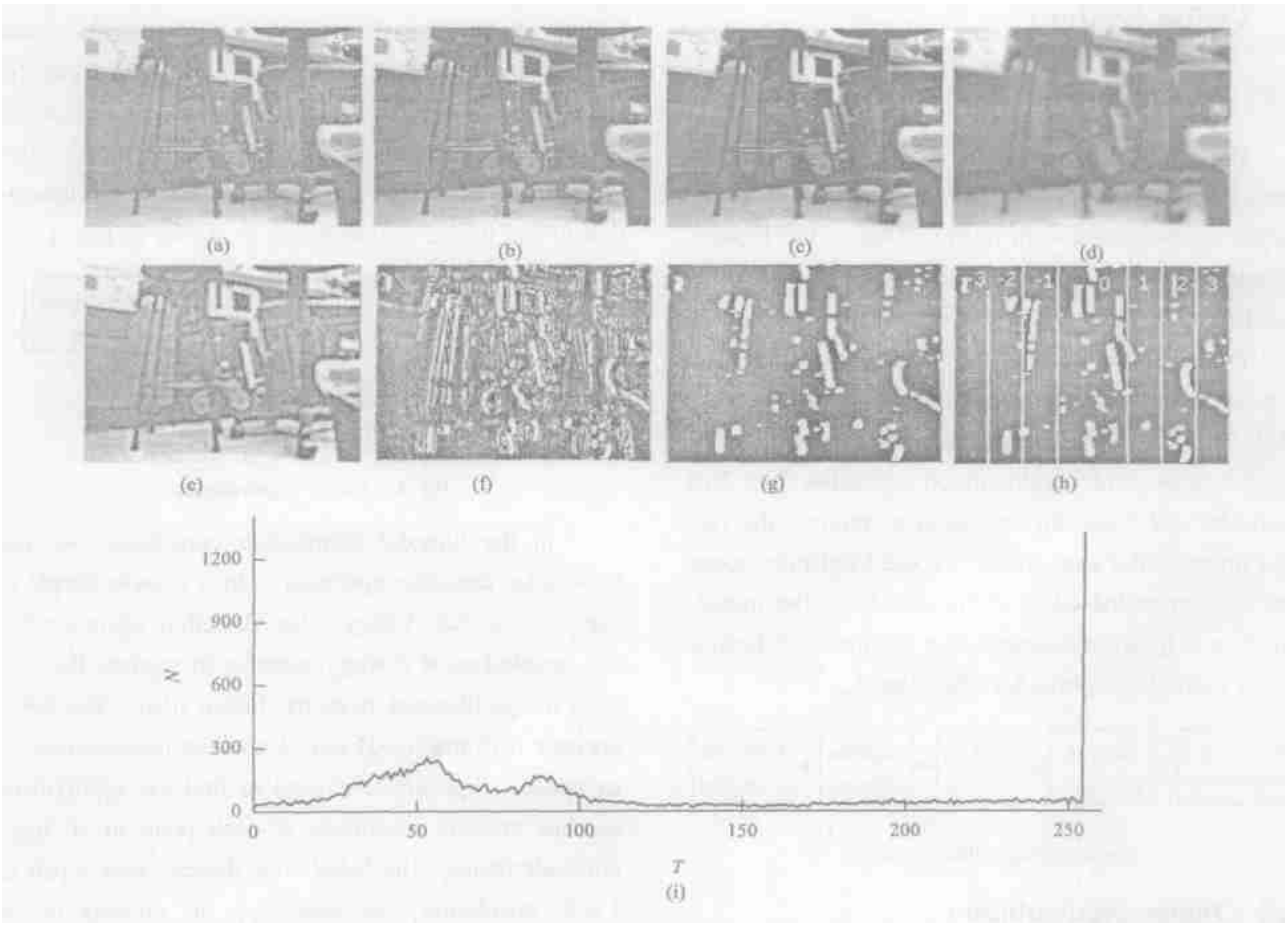

Fig. 5 Sampled multimodal image in vision system: (a) original image; (b) grey level image; (c) median filtered Image; (d) Gaussian filtered image; (e) Laplacian sharpened image; (f) edge detected image; (g) morphological close image; (h) vertical slices; (i) diagram of (e).

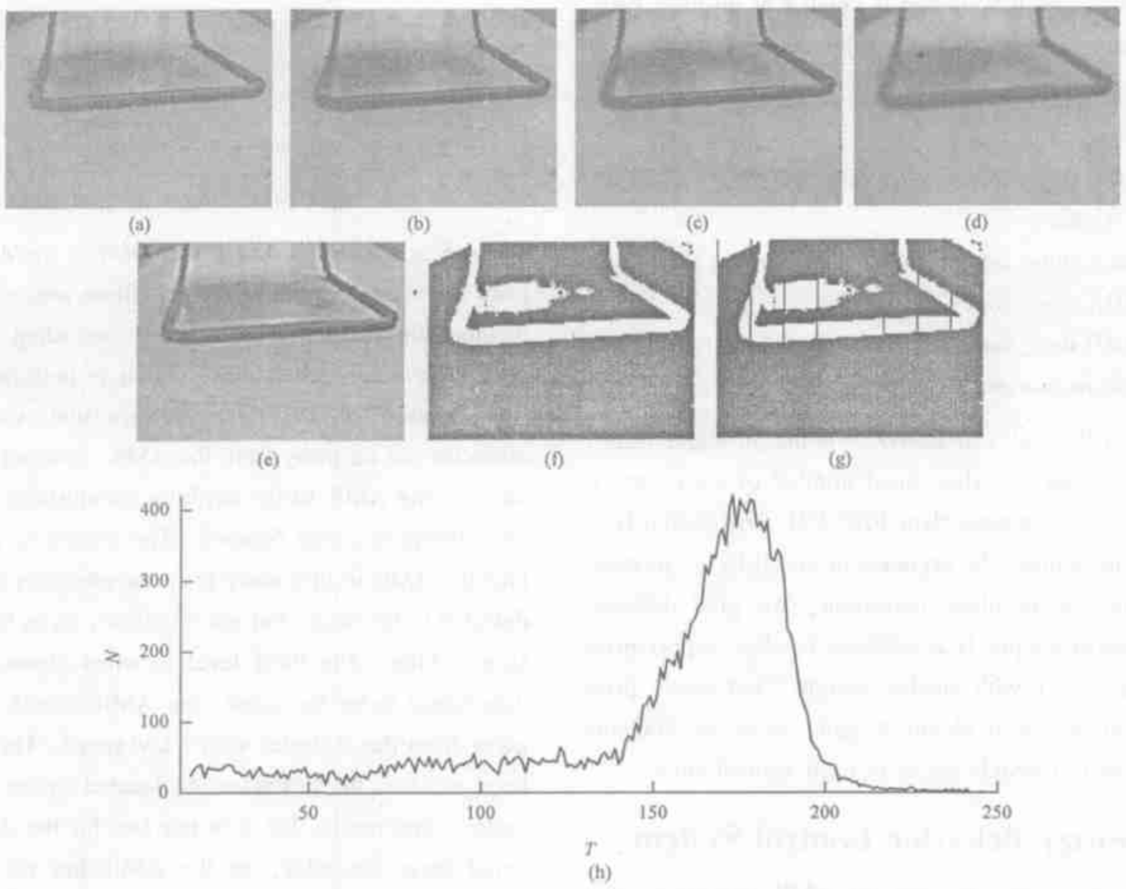

Fig. 6 Sampled single-modal image in vision system: (a) original image; (b) grey level image; (c) median filtered image; (d) Gaussian filtered image; (e) Laplacian sharpened image; (f) ($\mu - k\sigma$) thresholding image; (g) vertical slices; (h) diagram of (e).

2.3 Obstacle memory

Obstacle Memory is a memory array *OM*[36], which contains the latest information about all the obstacles around the AMR including the previous ones when AMR navigating in its environment. Each element represents the obstacle information in a 10° region; thus the obstacle memory just represents a full circle. When a segment image is generated, we divide it into seven vertical slices (−3, −2, −1, 0, 1, 2, 3), as shown in Fig. 6(h). Since we assume that the width of the AMR is twice as wide as the slice, the middle slice is designed to be twice as wide as the others, which is important for controlling moving forward behavior. Therefore, each current segment image only determines the obstacle information in a 70° front region, and the other 290° region is determined by history memory. If AMR reaches a region, it will generate obstacle information in the correspondent memory array; otherwise, that memory array will be set NULL. Therefore, we have

$$OM[i] = [dist, flag], \tag{5}$$

where $i = sfRobot.ath$ **div** 36, and *sfRobot.ath* is the current orientation of the AMR; *dist* is the closest distance between the obstacle and AMR in each vertical slice, which is mainly to control the AMR's translation velocity, and its value is not precise since it is computed via vision data, just like human beings can not get precise distance value through their eyes; *flag* represents whether the vertical slice *OM*[*i*] is an obstacle slice, which is the main factor to affect the steering angle.

While the AMR is moving in its environment, segment images are continuously being generated. Each time the new image is created at position (*ax*, *ay*, ath), where *ath* is the orientation of AMR. *OM* is reprocessed as follows

```
FOR $i = -3$ to 3
    IF sum-of-slice-pixels ( i ) > PIXNUM-THRESHOLD, THEN
        slice-index = ((ath div 36) + i + 36) mod 36,
        OM[ slice-index] . flag = 1,
        OM[ slice-index] . dist = image-height-$ay_{max}$-of-obstacle-point + image-bottom-to-AMR,
```

where *PIXNUM-THRESHOLD* is the threshold value; it means that the white pixel number of each vertical slice must be greater than *PIXNUM-THRESHOLD* in order to indicate the presence of an obstacle. Regarding the ground-plane constraint, we give different weights to the pixels at different heights: topper pixel (smaller *ay*) with smaller weight, and lower pixel (bigger *ay*) with bigger weight, when we compute the sum of obstacle pixels in each vertical slice.

3 Fuzzy Behavior Control System

Fuzzy logic is used for AMR navigation with obstacle avoidance. This is very important for the AMR to achieve all the actions to take. The objective here is to make a fuzzy behavior control system that generates velocity and steering angle for the AMR, and then leads to a fast and robust obstacle avoidance. The fuzzy system is shown in Fig.7, where *Vel* denotes the velocity of AMR and *Angl* denotes the turning angle of AMR.

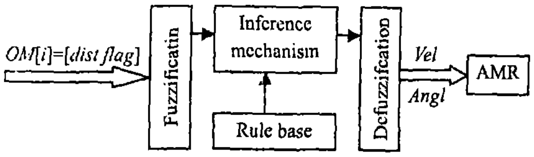

Fig. 7 Block diagram of the fuzzy system

3.1 Fuzzification

The fuzzification maps the crisp input values to the linguistic fuzzy terms with membership values between 0 and 1. Here four membership functions are used for the input of *dist* , which is shown in Fig.8,

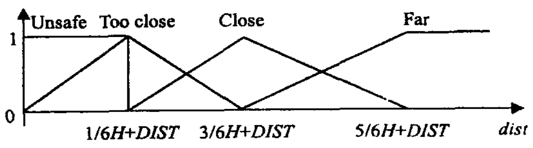

Fig. 8 Input (*dist*) membership functions

where *H* is the image height and *DIST* is the distance from the image bottom to AMR. These inputs of *dist* membership functions are defined according to the obstacle avoidance behavior, which is performed on four levels. The first level defines that when the obstacles are far away from the AMR, it is not necessary for the AMR to do anything for obstacle avoidance except to move forward. The second level enables the AMR to turn away from the obstacles that are detected to be close, but not too close, so to turn and slow a little. The third level is when obstacles are determined to be too close, the AMR should swerve away from the obstacles with a low speed. The fourth level is when the obstacles are located in the unsafe region, that means that it is too late for the AMR to avoid these obstacles, so the AMR has no choice except to move backward and rotate to find other safe passages.

As for the input of *flag* which mainly determines the steering angle, seven membership functions are designed shown in Fig. 9. When in ZERO, which means the front vertical slice 0 has no obstacle, the AMR does not need to change its steering angle and just moves forward. When slice −1 and slice −2 has no obstacles but slice 0 does, which is represented by PS (positive small), the AMR needs to turn left with a small angle to go into this region to avoid obstacles. When there are obstacles in slice 0 and slice −1, but no ones in slice −2 and slice −3, which is represented by PM (positive middle), we should let the AMR turn a middle angle go into this region. If all of the above conditions do not exist, which is represented by PT (positive trap), the AMR should move backward and rotate to find other ways. The same occurs to the negative side.

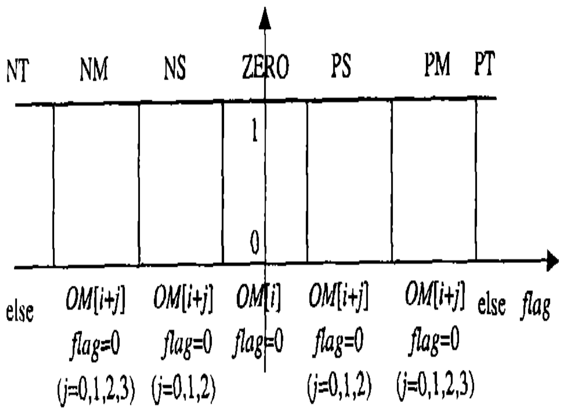

Fig. 9 Input (*flag*) membership functions

3.2 Inference mechanism

The inference mechanism is responsible for the decision making in the control system using approximation reasoning. It is used to combine the fuzzy "IF-THEN" with the fuzzy rule base, and to convert the input information into the output membership functions. The inference mechanism emulates an expert's decision-making knowledge in interpreting and applying knowledge about how to perform the control tasks. The knowledge can be implemented as a fuzzy rule base which stores the rules governing the input-output relationship. The control rules are formulated in Table 1 and Table 2.

Table 1 Rules for the velocity

flag	*dist*			
	Unsafe	Too close	Close	Far
ZERO	Positive	Positive	Positive fast	Positive fast
PS	Zero	Positive	Positive fast	Positive fast
PM	Negative	Zero	Positive	Positive fast
PT	Negative	Negative	Zero	Positive

Table 2 Rules for the steering angle ("to turn left" and "to turn right" are the same)

flag	*dist*			
	Unsafe	Too close	Close	Far
ZERO	Zero	Zero	Zero	Zero
PS	Small	Small	Small	Small
PM	Large	Large	Large	Large
PT	Zero	Zero	Zero	Zero

Obstacle avoidance which is designed to avoid unexpected obstacles uses the obstacle memory including current and previous obstacle information to determine which direction should turn, right or left. If there are more obstacles in the left semicircle region including the obstacles in history than the right, the AMR should turn right, and vice versa. After determining to turn right or left, we use this fuzzy system to produce the steering angle and translation velocity. There will be some error in the motors during the navigation, which leads to some turn error for the AMR. Therefore, to keep the AMR moving forward when there is no obstacles in the front passage, we adjust both the pan and tilt angle of the camera to the maximum angle ($\alpha_{\text{pan-max}} = \pm 20^{\circ}$, $\alpha_{\text{tilt-max}} = \pm 95^{\circ}$) to see if there is any obstacle in unsafe region in the right side or the left side. The zoom of the camera is temporarily not used. If any obstacle is detected in the right side, the AMR turn left a small angle, and vice versa. If there are obstacles in both sides, the AMR does not turn a bit and has a try to move ahead since moving forward has the higher priority.

3.3 Defuzzification

The defuzzification maps the fuzzy output from the inference mechanism to a crisp signal. The "center of gravity" (COG) defuzzification method is used to combine the recommendations represented by the implied fuzzy sets from all the rules. Let b_i denote the center of the membership function of the consequent of rule i, and $\int \mu(i)$ denote the area under the membership function $\mu(i)$. The COG method computes the crisp output μ_{crisp} by

$$\mu_{\text{crisp}} = \frac{\sum b_i \int \mu(i)}{\sum_i \int \mu(i)}, \tag{6}$$

and the output membership functions are shown in Fig. 10.

4 Experiments and Results

The system described in this paper was implemented on the Pioneer 1 mobile robot (shown in Fig. 1) and the system architecture is shown in Fig. 11. The AMR communicates with the workstation over two links, a video link for the transmission of image data and RF link for control commands. The received image is processed on the workstation and the resulting control commands

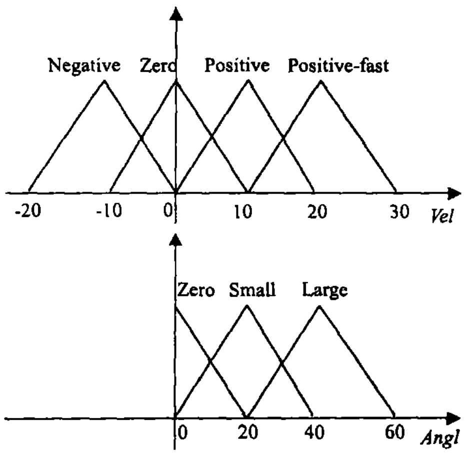

Fig. 10 Output membership functions for *Vel* and *Angl*

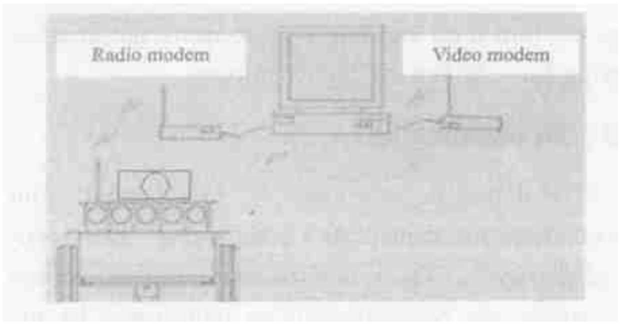

Fig. 11 System architecture in pioneer 1

are sent to the AMR over the RF link. The image processing time from an original frame with true color to obstacle memory and behavior operation time are dependent on the workstation processor. In this experiment, we use a Pentium 3 ATX 500 MHz computer as workstation. The capture frame rate is 10 frame per second and the frame size is 160×120 pixels. In this way, the image processing time is shown in Table 3. It is seen that these processing and operation time is sufficient for real-time requirements. Some experiment results are shown in Fig. 12.

Table 3 Image processing time and behavior operation time per frame ms

Method	Capture a frame → thresholding image (or closing image)	Segmentation→ Obstacle memory	All kinds of behaviors operation time
($\mu - k\sigma$) Thresholding technique	20—30	0—10	10—15
Morphological closing based on Sobel edge detection	31—52	0—10	10—15

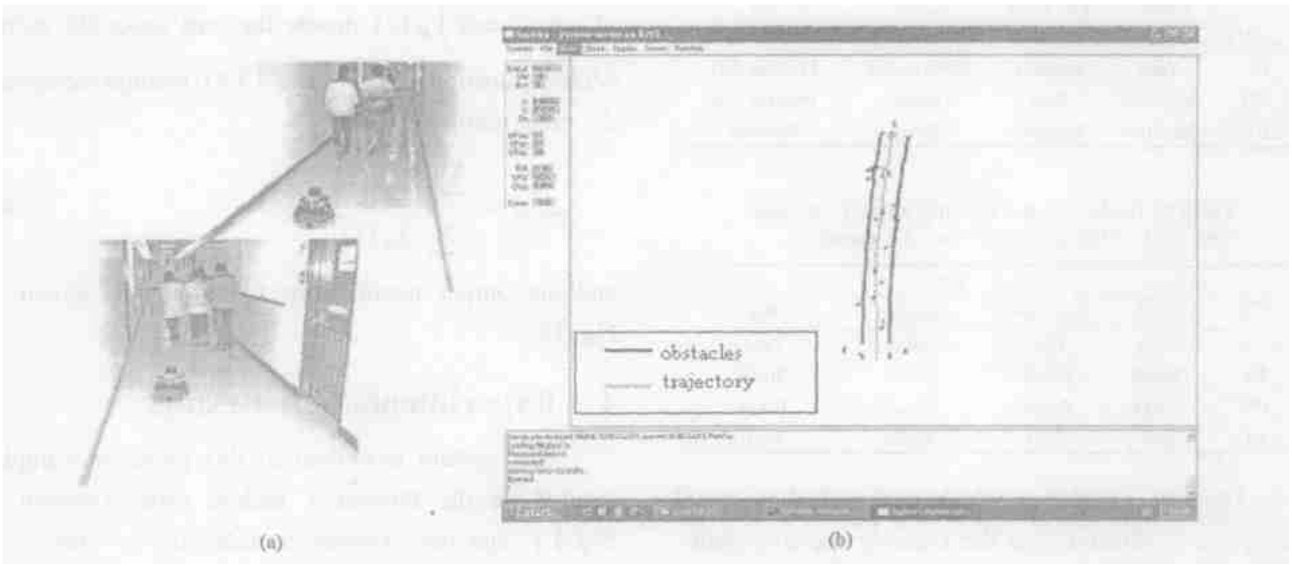

Fig. 12 Experimental sample: (a) AMR in experiment; (b) AMR's trajectory

5 Conclusions and Future Work

The system in this paper consists of a vision subsystem and a fuzzy behavior control subsystem. In the vision subsystem, the method of combining morphological closing operation based on Sobel edge detection operation and ($\mu - k\sigma$) thresholding technique has proven to be robust for lighting and ground variations and computationally simple. In addition, the system can remember historical obstacle information, so in case that current image is totally damaged, the AMR still can correctly navigate based on its obstacle memory, just like that human beings can remember previous obstacles when moving, and if in a sudden and temporary darkness, they can use this memory to avoid the obstacles. The developed control system is a subsumption architecture realized with fuzzy logic. Each behavior is computationally simple and fault tolerant since it does not rely upon accurate sensing data. Multiple behaviors are solely based on the obstacle memory and run in priority coordinator. Based on the fast processing and reactive behavior, the AMR can tackle some dynamic environments. Furthermore, a new module can be added to the system without any modification of the existing modules. This flexibility is crucial for the future research. Future work involves improving the segmentation algorithm to detect obstacles more accurately and robust, and to solve the vision-based localization problem.

References

[1] Wang Z, Admadabadi M N, Nakano E, et al. A multiple robot system for cooperative object transportation with various requirement on task performing[A]. In: *Proc. of ICRA 99* [C]. Detroit, MI USA, 1999: 1226—1233.

[2] Ohya A, Kosaka A, Kak A. Vision-based navigation of mobile robot with obstacle avoidance by single camera vision and ultrasonic sensing[J]. *IEEE Transactions on Robotics and Automation*, 1998, 14(6): 969—978.

[3] Lorigo L M, Brooks R A, Grimson W E L. Visually-guided obstacle avoidance in unstructured environments[A]. In: *IEEE International Conference on Intelligent Robots and Systems*[C]. Grenoble, France, 1997: 373—379.

[4] Potts J J. Real-time vision-based autonomous robot navigation[D]. Illinois: University of Illinois at Urbana-Champaign, 1999.

[5] Miin T C, Braunl, T, Zaknich A. Visually-guided obstacle avoidance[A]. In: *Proc. of ICONIP 99*[C]. Perth, WA Australia, 1999: 650—655.

[6] Maja J M, Takahashi T, Wang Z D, et al. Real-time obstacle avoidance algorithm for visual navigation[A]. *In: IEEE International Conference on Intelligent Robots and Systems*[C]. Takamatsu, Japan, 2000: 925—930.

[7] TanakaK. Detecting collision-free paths by observing walking people[A]. In: *IEEE International Conference on Intelligent Robots and Systems*[C]. Lausanne, Switzerland, Sep 30-Oct 4, 2002: 55—60.

[8] Ku C H, Tsai W H. Obstacle avoidance in person following for vision-based autonomous land vehicle guidance using vehicle location estimation and quadratic pattern classifier[J]. *IEEE Transaction on Industrial Electronics*, 2001, 48(1): 205—215.

[9] Brooks R A. A robust layered control system for a mobile robot[J]. *IEEE Journal of Robotics and Automation*, 1986, 2(1): 14—23.

[10] Li H, Yang S X. A behavior-based mobile robot with a visual landmark-recognition system[J]. *IEEE/ASME Transactions on Mechatronics*, 2003, 8(3): 390—400.

[11] Dadios E P, Maravillas O A Jr. Cooperative mobile robots with obstacle and collision avoidance using fuzzy logic[A]. In: *Proc. of 2002 IEEE Int. Symposium on Intelligent Control*[C]. Vancouver, Canada, Oct 27—30, 2002: 75—80.

[12] Cho J T, Nam B H. A study on the fuzzy control navigation and the obstacle avoidance of mobile robot using camera [A]. In: *Proc. of 2000 IEEE Int. Conf. on Systems, Man, and Cybernetics*[C]. Nashville, TN USA, 2000: 2993—2997.

[13] Forsyth P. *Computer vision: A modern approach*[M]. Upper Saddle River, N. J.: Prentice Hall, 2003.

[14] Canny J F. Finding edges and lines in images[D]. Massachusetts: MIT, 1983.

[15] Jahne B, Horst H. *Computer vision and applications: A guide for students and practitioners*[M]. San Diego: Academic Press, 2000.